K 6

```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR      FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR      FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR      FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR      FFFFFFFFFFFF
EEE              RRR   RRR         FFF
EEE              RRR   RRR         FFF
EEE              RRR   RRR         FFF
EEE              RRR      RRR      FFF
EEE              RRR      RRR      FFF
EEE              RRR      RRR      FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

```
_$

Va
--
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
```

```
**FILE**ID**DUTUDRIVR
```

```
DDDDDDDD    UU      UU  TTTTTTTTTT  UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  RRRRRRRR
DDDDDDDD    UU      UU  TTTTTTTTTT  UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  RRRRRRRR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR      RR      II      VV      VV  RR      RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR      RR      II      VV      VV  RR      RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR      RR      II      VV      VV  RR      RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR      RR      II      VV      VV  RR      RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RRRRRRRR        II      VV      VV  RRRRRRRR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RRRRRRRR        II      VV      VV  RRRRRRRR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR  RR          II      VV      VV  RR  RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR  RR          II      VV      VV  RR  RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR    RR        II        VV  VV    RR    RR      ....
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR    RR        II        VV  VV    RR    RR      ....
DDDDDDDD    UUUUUUUUUU      TT      UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII        VV      RR      RR    ....
DDDDDDDD    UUUUUUUUUU      TT      UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII        VV      RR      RR    ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
C
C Version:      'V04-000'
C
C****************************************************************************
C*                                                                          *
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
C*  ALL RIGHTS RESERVED.                                                    *
C*                                                                          *
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
C*  TRANSFERRED.                                                            *
C*                                                                          *
C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
C*  CORPORATION.                                                            *
C*                                                                          *
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
C*                                                                          *
C*                                                                          *
C****************************************************************************
C

c       Author  Brian Porter                    Creation date   10-FEB-1982

c++
c       Functional description:
c
c       This module displays entries logged by MSCP disks (dudriver) and
c       and MSCP tapes (tudriver).
c
c       Modified by:
c
C       V03-007 EAD0200         Elliott A. Drayton              23-Jul-1984
C               Added code to mark the begining of the intervening entries.
C
C       V03-006 SAR0272         Sharon A. Reynolds      18-Jun-1984
C               - Re-structured and re-named the routines in this
C               module to handle disk or tape MSCP entries for the
C               addition of TMSCP support.
C
C       V03-005 SAR0197         Sharon A. Reynolds,     20-Feb-1984
C               Added an SYE update that:
C               - Removed 'invalid mscp command end message'.
C
C       V03-004 SAR0157         Sharon A. Reynolds,     12-Oct-1983
C               Added an SYE update that:
C               - adds an extra arguement to the 'dudriver_mscp_dispatcher'
C                 routine.
C               - adds an extra arguement to the call for the
C                 'dudriver_mscp_dispatcher' routine.
C               - adds an extra arguement to the calls for several
C                 routines that reside in 'mscp.for'.
```

```
C
C       V03-003 SAR0072         Sharon A. Reynolds,     20-Jun-1983
C               Changed the carriage control in the 'format' statements
C               for use with ERF.
C
c       v03-002 BP0002          Brian Porter,           08-FEB-1983
c               Corrected argument list to erllogmsg2.
c
c       v03-001 BP0001          Brian Porter,           19-APR-1982
c               Made changes to accomodate invalid command mscp messages.
c**
c--

        Subroutine DISK_TAPE_DRVR_MSCP_DISPATCHER (lun,option,reccnt,
        1 mount_flag_and_label,record_length,queue_count)


        include 'src$:msghdr.for /nolist'
        include 'src$:emblmdef.for /nolist'
        include 'src$:embspdef.for /nolist'


        byte            lun

        character*1     option

c  This value RECCNT is not the record number of the entry just read from the
c  errlog.sys file it is the value which was saved in the queue when this
c  routine is called by _DQ.
        integer*4       reccnt
c
        integer*4       mount_flag_and_label
        integer*4       record_length
        Integer*4       queue_count
        Integer*4       packet_length

        byte            mslg$b_format
        equivalence     (emb(46),mslg$b_format)


        if (emb$w_hd_entry .eq. 100) then               ! Logmessage entry

C
C Determine whether to output the long or short header and call
C the appropriate routine.
C
        If (queue_count .EQ. 1) then

        Call FRCTOF (lun)
        Call HEADER2 (lun,reccnt)
        Else

        Call HEADER3 (lun,reccnt)
        Endif

        Call LOGGER (lun,'ERL$LOGMESSAGE ENTRY')

        Call DHEAD3 (lun,'I/O',emb$b_lm_namlng,emb$t_lm_name,emb$w_lm_unit,
        1 mount_flag_and_label)

        Packet_length = record_length - 39

        if (mslg$b_format .eq. 0) then          ! Controller error

        if (option .eq. 'S') then
        Call MSLG$K_CNT_ERR (lun,packet_length)
        endif

        else if (mslg$b_format .eq. 1) then     ! Memory access error

        if (option .eq. 'S') then
        Call MSLG$K_BUS_ADDR (lun,packet_length)
        endif

        else if (               ! Disk/tape transfer error
        1 mslg$b_format .eq. 2  ! mslg$k_disk_trn
        1 .OR.
        1 mslg$b_format .EQ. 5  ! mslg$k_tape_trn
        1 ) then

        if (option .eq. 'S') then
        Call DISK_TAPE_TRANSFER_ERRORS (lun,packet_length)
        endif

        else if (               ! SDI/STI errors
        1 mslg$b_format .eq. 3  ! Disk SDI comm error - mslg$k_sdi
        1 .OR.
        1 mslg$b_format .EQ. 6  ! Tape STI comm or cmd failure - mslg$k_sti_err
        1 .OR.
        1 mslg$b_format .EQ. 7  ! Tape STI Drive Error Log - mslg$k_sti_del
        1 .OR.
        1 mslg$b_format .EQ. 8  ! Tape STI Formatter Error Log - mslg$k_sti_fel
        1 ) then

        if (option .eq. 'S') then
        Call SDI_STI_ERRORS (lun,packet_length)
        endif

        else if (mslg$b_format .eq. 4) then     ! Small Disk error

        if (option .eq. 'S') then
        Call MSLG$K_SML_DSK (lun,packet_length)
        endif

        else
C
C Unknown format type, call a routine that will decode/output the header
C  information and dump the rest of the packet in a hex longword format.
C
        Call ERLLOGMSG2 (lun,record_length)
        endif

        else if (emb$w_hd_entry .eq. 99) then   ! Logstatus entry

```

```
      C
      C Determine whether to output the long or short header and call
      C the appropriate routine.
      C
              If (queue_count .EQ. 1) then
              Call FRCTOF (lun)
              Call HEADER2 (lun,reccnt)

              Else
              Call HEADER3 (lun,reccnt)
              Endif

              Call LOGGER (lun,'ERL$LOGSTATUS ENTRY')

              Call DHEAD3 (lun,'I/O',emb$b_sp_namlng,emb$t_sp_name,emb$w_sp_unit,
              1 mount_flag_and_label)

              Call ERLLOGSTS2 (lun)
              endif

              return
              end
```

```
PROGRAM SECTIONS

    Name                                      Bytes   Attributes

  0 $CODE                                       426   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                       45   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                      164   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 EMB                                         512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                      1147


ENTRY POINTS

    Address   Type  Name

  0-00000000        DISK_TAPE_DRVR_MSCP_DISPATCHER


VARIABLES

    Address   Type  Name                                   Address   Type  Name

  3-00000010  L*1   EMB$B_LM_CLASS                       3-00000014  L*1   EMB$B_LM_NAMLNG
  3-00000011  L*1   EMB$B_LM_TYPE                        3-00000010  L*1   EMB$B_SP_CLASS
  3-00000040  L*1   EMB$B_SP_NAMLNG                      3-00000011  L*1   EMB$B_SP_TYPE
  3-00000000  I*4   EMB$L_HD_SID                         3-00000014  I*4   EMB$L_SP_BCNT
  3-00000038  I*4   EMB$L_SP_CHAR                        3-0000003C  I*4   EMB$L_SP_CMDREF
  3-00000020  I*4   EMB$L_SP_IOSB1                       3-00000024  I*4   EMB$L_SP_IOSB2
  3-00000018  I*4   EMB$L_SP_MEDIA                       3-0000002C  I*4   EMB$L_SP_OPCNT
  3-00000034  I*4   EMB$L_SP_OWNUIC                      3-0000001C  I*4   EMB$L_SP_RQPID
  3-00000015  CHAR EMB$T_LM_NAME                         3-00000041  CHAR EMB$T_SP_NAME
  3-00000004  I*2  EMB$W_HD_ENTRY                        3-0000000E  I*2  EMB$W_HD_ERRSEQ
  3-00000024  I*2  EMB$W_LM_MSGTYP                       3-00000012  I*2  EMB$W_LM_UNIT
  3-00000012  I*2  EMB$W_SP_BOFF                         3-00000030  I*2  EMB$W_SP_ERRCNT
  3-00000028  I*2  EMB$W_SP_FUNC                         3-00000032  I*2  EMB$W_SP_STS
  3-0000002A  I*2  EMB$W_SP_UNIT                        AP-00000004@ L*1  LUN
 AP-00000010@ I*4  MOUNT_FLAG_AND_LABEL                  3-0000002E  L*1  MSLG$B_FORMAT
 AP-00000008@ CHAR OPTION                                2-00000000  I*4  PACKET_LENGTH
 AP-00000018@ I*4  QUEUE_COUNT                          AP-0000000C@ I*4  RECCNT
 AP-00000014@ I*4  RECORD_LENGTH


ARRAYS

    Address  Type  Name                                          Bytes  Dimensions

  3-00000000  L*1  EMB                                             512  (0:511)
  3-00000026  L*1  EMB$B_LM_MSGTXT                                 460  (460)
  3-00000006  I*4  EMB$Q_HD_TIME                                     8  (2)


FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name                                Type  Name                                Type  Name

        DHEAD3                                    DISK_TAPE_TRANSFER_ERRORS                 ERLLOGMSG2
        ERLLOGSTS2                                FRCTOF                                    HEADER2
        HEADER3                                   LOGGER                                    MSLG$K_BUS_ADDR
        MSLG$K_CNT_ERR                            MSLG$K_SML_DSK                            SDI_STI_ERRORS
```

```



c++
c       Since mscp error log entries are delivered from the port via
c       the datagram service it is possible for them to be delivered
c       out of sequence or duplicated.  It is the responsibility of
c       this queue to collect all entries containing the same command
c       reference for a given cpu together.  They are placed in order
c       of error log entry type.
c
c       The format of the elements are as follows
c
c       +-----------------------------------+
c       :              flink1               :
c       +-----------------------------------+
c       :              blink1               :
c       +-----------------------------------+
c       :            logging sid            :
c       +-----------------------------------+
c       :  root command reference flink     :
c       +-----------------------------------+
c       :  root command reference blink     :
c       +-----------------------------------+
c       :  command reference entry count    :
c       +-----------------------------------+
c
c       +-----------------------------------+
c       :              flink2               :
c       +-----------------------------------+
c       :              blink2               :
c       +-----------------------------------+
c       :     command reference number      :
c       +-----------------------------------+
c       :    root_emb$$w_hd_entry_flink     :
c       +-----------------------------------+
c       :    root_emb$$w_hd_entry_blink     :
c       +-----------------------------------+
c       :      emb$$w_hd_entry count        :
c       +-----------------------------------+
c
c       +-----------------------------------+
c       :              flink3               :
c       +-----------------------------------+
c       :              blink3               :
c       +-----------------------------------+
c       :          emb$$w_hd_entry          :
c       +-----------------------------------+
c       :      error log record number      :
c       +-----------------------------------+
c       :  error log record size (bytes)    :
c       +-----------------------------------+
c       :                                   :
c       +--                               --+
c       .                                   .
c       .         error log record          .
c       .                                   .
```

```
c       +--                               --+
c       :                                   :
c       +-----------------------------------+
c       :           mounted flag            :
c       +-----------------------------------+
c       :                                   :
c       +--                               --+
c       :   mounted volume label (if any)   :
c       +--                               --+
c       :                                   :
c       +-----------------------------------+
c--



        Subroutine DISK_TAPE_DRIVERS_MSCP_Q (record_length,reccnt,
        1 search_command_reference_number)


        include 'src$:msghdr.for /nolist'
        include 'src$:emblmdef.for /nolist'
        include 'src$:embspdef.for /nolist'



        byte                lun

        character*1         option

        integer*4           record_length
        integer*4           reccnt
        integer*4           search_command_reference_number
        integer*4           buffer(2)
        logical*4           lib$get_vm
        integer*4           insert_blink
        integer*4           logging_sid_entry_address
        integer*4           command_reference_entry_address
        integer*4           emb$$w_hd_entry_address
        integer*4           root_logging_sid_flink
        integer*4           root_logging_sid_blink

        equivalence         (buffer(1),root_logging_sid_flink)
        equivalence         (buffer(2),root_logging_sid_blink)

        integer*4           logging_sid_entry_count
        data                logging_sid_entry_count /0/

        integer*4           buffer1(6)
        integer*4           flink1
        integer*4           blink1
        integer*4           logging_sid
        integer*4           root_command_reference_flink
        integer*4           root_command_reference_blink
        integer*4           command_reference_entry_count

        equivalence         (buffer1(1),flink1)
        equivalence         (buffer1(2),blink1)
```

```
            equivalence      (buffer1(3),logging_sid)
            equivalence      (buffer1(4),root_command_reference_flink)
            equivalence      (buffer1(5),root_command_reference_blink)
            equivalence      (buffer1(6),command_reference_entry_count)

            integer*4        buffer2(6)
            integer*4        flink2
            integer*4        blink2
            integer*4        command_reference_number
            integer*4        root_emb$$w_hd_entry_flink
            integer*4        root_emb$$w_hd_entry_blink
            integer*4        emb$$w_hd_entry_count

            equivalence      (buffer2(1),flink2)
            equivalence      (buffer2(2),blink2)
            equivalence      (buffer2(3),command_reference_number)
            equivalence      (buffer2(4),root_emb$$w_hd_entry_flink)
            equivalence      (buffer2(5),root_emb$$w_hd_entry_blink)
            equivalence      (buffer2(6),emb$$w_hd_entry_count)

            integer*4        buffer3(5)
            integer*4        flink3
            integer*4        blink3
            integer*4        emb$$w_hd_entry
            integer*4        error_log_record_number
            integer*4        error_log_record_length

            equivalence      (buffer3(1),flink3)
            equivalence      (buffer3(2),blink3)
            equivalence      (buffer3(3),emb$$w_hd_entry)
            equivalence      (buffer3(4),error_log_record_number)
            equivalence      (buffer3(5),error_log_record_length)


            if (logging_sid_entry_count .eq. 0) then

            root_logging_sid_flink = %loc(root_logging_sid_flink)
            root_logging_sid_blink = root_logging_sid_flink
            endif

            logging_sid_entry_address = root_logging_sid_flink

            do 100,i = 1,logging_sid_entry_count

            call movc3 (%val(24),%val(logging_sid_entry_address),buffer1)

            if (logging_sid .eq. emb$l_hd_sid) then

      10    command_reference_entry_address = root_command_reference_flink

            do 90,j = 1,command_reference_entry_count

            call movc3 (%val(24),%val(command_reference_entry_address),buffer2)

            if (command_reference_number .eq. search_command_reference_number)
            1 then

```

```
25      insert_blink = root_emb$$w_hd_entry_blink

        if (emb$$w_hd_entry_count .ne. 0) then

        call movc3 (%val(12),%val(root_emb$$w_hd_entry_blink),buffer3)

        if (emb$$w_hd_entry .lt. emb$w_hd_entry) then

        insert_blink = blink3
        endif
        endif

        call movc5 (%val(0),,%val(0),%val(20),buffer3)

        if (lib$get_vm((20+record_length+16),emb$$w_hd_entry_address)) then

        call insque (%val(emb$$w_hd_entry_address),%val(insert_blink))

        emb$$w_hd_entry = emb$w_hd_entry

        error_log_record_number = reccnt

        error_log_record_length = record_length

        call movc3 (%val(12),emb$$w_hd_entry,
        1 %val(emb$$w_hd_entry_address + 8))

        call movc3 (%val(record_length),emb,%val(emb$$w_hd_entry_address + 20))

        call movl (-1,%val(emb$$w_hd_entry_address+20+record_length))

        if (emb$w_hd_entry .eq. 100) then

        call get_current_label (3,emb$l_hd_sid,emb$b_lm_namlng,emb$t_lm_name,
        1 emb$w_lm_unit,%val(emb$$w_hd_entry_address+20+record_length+4),*30)

        else if (emb$w_hd_entry .eq. 99) then

        call get_current_label (3,emb$l_hd_sid,emb$b_sp_namlng,emb$t_sp_name,
        1 emb$w_sp_unit,%val(emb$$w_hd_entry_address+20+record_length+4),*30)
        endif

        call movl (emb$$w_hd_entry_address+20+record_length+4,
        1 %val(emb$$w_hd_entry_address+20+record_length))

30      emb$$w_hd_entry_count = emb$$w_hd_entry_count + 1

        call movl (emb$$w_hd_entry_count,
        1 %val(command_reference_entry_address + 20))
        endif

        return
        endif

        command_reference_entry_address = flink2

90      continue
```

```
                call movc5 (%val(0),,%val(0),%val(24),buffer2)

                if (lib$get_vm(24,command_reference_entry_address)) then

                call insque (%val(command_reference_entry_address),
                1 %val(root_command_reference_blink))

                command_reference_number = search_command_reference_number

                root_emb$$w_hd_entry_flink = command_reference_entry_address + 12

                root_emb$$w_hd_entry_blink = root_emb$$w_hd_entry_flink

                call movc3 (%val(16),command_reference_number,
                1 %val(command_reference_entry_address + 8))

                command_reference_entry_count = command_reference_entry_count + 1

                call movl (command_reference_entry_count,
                1 %val(logging_sid_entry_address + 20))

                goto 25
                endif

                return
                endif

                logging_sid_entry_address = flink1

        100     continue

                call movc5 (%val(0),,%val(0),%val(24),buffer1)

                if (lib$get_vm(24,logging_sid_entry_address)) then

                call insque (%val(logging_sid_entry_address),
                1 %val(root_logging_sid_blink))

                logging_sid = emb$l_hd_sid

                root_command_reference_flink = logging_sid_entry_address + 12

                root_command_reference_blink = root_command_reference_flink

                call movc3 (%val(16),logging_sid,%val(logging_sid_entry_address + 8))

                logging_sid_entry_count = logging_sid_entry_count + 1

                goto 10
                endif

                return


                entry DISK_TAPE_DRIVERS_MSCP_DQ (lun,option)
                logging_sid_entry_address = root_logging_sid_flink

                If (logging_sid_entry_count .GT. 0) then
                    Write (lun, 9000)
        9000        Format (////////////////////
                1'      B E G I N I N G   O F   I N T E R V E N I N G   E N T R I E S')
                Endif

                do 150,i = 1,logging_sid_entry_count

                call movc3 (%val(24),%val(logging_sid_entry_address),buffer1)

                command_reference_entry_address = root_command_reference_flink

                do 200,j = 1,command_reference_entry_count

                call movc3 (%val(24),%val(command_reference_entry_address),buffer2)

                emb$$w_hd_entry_address = root_emb$$w_hd_entry_flink

                do 250,k = 1,emb$$w_hd_entry_count

                call movc3 (%val(20),%val(emb$$w_hd_entry_address),buffer3)

                call movc5 (%val(0),,%val(0),%val(512),emb)

                call movc3 (%val(error_log_record_length),
                1 %val(emb$$w_hd_entry_address + 20),emb)

                call DISK_TAPE_DRVR_MSCP_DISPATCHER (lun,option,
                1 error_log_record_number,
                1 %val(emb$$w_hd_entry_address+20+error_log_record_length),
                1 error_log_record_length,k)

                emb$$w_hd_entry_address = flink3

        250     continue

                command_reference_entry_address = flink2

        200     continue

                logging_sid_entry_address = flink1

        150     continue

                return

                end
```

```
DISK_TAPE_DRIVERS_MSCP_Q                              16-Sep-1984 00:20:36    VAX-11 FORTRAN V3.4-56          Page  12
                                                       5-Sep-1984 13:54:29    DISK$VMSMASTER:[ERF.SRC]DUTUDRIVR.FOR;1

PROGRAM SECTIONS

   Name                                   Bytes   Attributes

 0 $CODE                                    900   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
 1 $PDATA                                    98   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
 2 $LOCAL                                   548   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
 3 EMB                                      512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

   Total Space Allocated                   2058


ENTRY POINTS

   Address  Type  Name                                Address  Type  Name

 0-00000290       DISK_TAPE_DRIVERS_MSCP_DQ         0-00000000       DISK_TAPE_DRIVERS_MSCP_Q


VARIABLES

   Address  Type  Name                                Address  Type  Name

  2-00000030  I*4  BLINK1                            2-00000018  I*4  BLINK2
  2-00000004  I*4  BLINK3                            2-00000054  I*4  COMMAND_REFERENCE_ENTRY_ADDRESS
  2-00000040  I*4  COMMAND_REFERENCE_ENTRY_COUNT     2-0000001C  I*4  COMMAND_REFERENCE_NUMBER
  2-00000008  I*4  EMB$SW_AD_ENTRY                   2-00000058  I*4  EMB$SW_AD_ENTRY_ADDRESS
  2-00000028  I*4  EMB$SW_HD_ENTRY_COUNT             3-00000010  L*1  EMB$B_LM_CLASS
  3-00000014  L*1  EMB$B_LM_NAMLNG                   3-00000011  L*1  EMB$B_LM_TYPE
  3-00000010  L*1  EMB$B_SP_CLASS                    3-00000040  L*1  EMB$B_SP_NAMLNG
  3-00000011  L*1  EMB$B_SP_TYPE                     3-00000000  I*4  EMB$L_HD_SID
  3-00000014  I*4  EMB$L_SP_BCNT                     3-00000038  I*4  EMB$L_SP_CHAR
  3-0000003C  I*4  EMB$L_SP_CMDREF                   3-00000020  I*4  EMB$L_SP_IOSB1
  3-00000024  I*4  EMB$L_SP_IOSB2                    3-00000018  I*4  EMB$L_SP_MEDIA
  3-0000002C  I*4  EMB$L_SP_OPCNT                    3-00000034  I*4  EMB$L_SP_OWNUIC
  3-0000001C  I*4  EMB$L_SP_RQPID                    3-00000015  CHAR EMB$T_LM_NAME
  3-00000041  CHAR EMB$T_SP_NAME                     3-00000004  I*2  EMB$W_HD_ENTRY
  3-0000000E  I*2  EMB$W_HD_ERRSEQ                   3-00000024  I*2  EMB$W_LM_MSGTYP
  3-00000012  I*2  EMB$W_LM_UNIT                     3-00000012  I*2  EMB$W_SP_BOFF
  3-00000030  I*2  EMB$W_SP_ERRCNT                   3-00000028  I*2  EMB$W_SP_FUNC
  3-00000032  I*2  EMB$W_SP_STS                      3-0000002A  I*2  EMB$W_SP_UNIT
  2-00000010  I*4  ERROR_LOG_RECORD_LENGTH           2-0000000C  I*4  ERROR_LOG_RECORD_NUMBER
  2-0000002C  I*4  FLINK1                            2-00000014  I*4  FLINK2
  2-00000000  I*4  FLINK3                            2-00000060  I*4  I
  2-0000004C  I*4  INSERT_BLINK                      2-00000064  I*4  J
  2-00000068  I*4  K                                 2-00000034  I*4  LOGGING_SID
  2-00000050  I*4  LOGGING_SID_ENTRY_ADDRESS         2-0000005C  I*4  LOGGING_SID_ENTRY_COUNT
 AP-00000004@ L*1  LUN                              AP-00000008@ CHAR OPTION
 AP-00000008@ I*4  RECCNT                           AP-00000004@ I*4  RECORD_LENGTH
  2-0000003C  I*4  ROOT_COMMAND_REFERENCE_BLINK      2-00000038  I*4  ROOT_COMMAND_REFERENCE_FLINK
  2-00000024  I*4  ROOT_EMB$SW_AD_ENTRY_BLINK        2-00000020  I*4  ROOT_EMB$SW_AD_ENTRY_FLINK
  2-00000048  I*4  ROOT_LOGGING_SID_BLINK            2-00000044  I*4  ROOT_LOGGING_SID_FLINK
 AP-0000000C@ I*4  SEARCH_COMMAND_REFERENCE_NUMBER
```

```
                                                                         J  1
DISK_TAPE_DRIVERS_MSCP_Q                                              16-Sep-1984 00:20:36    VAX-11 FORTRAN V3.4-56              Page  13
                                                                       5-Sep-1984 13:54:29    DISK$VMSMASTER:[ERF.SRC]DUTUDRIVR.FOR;1

ARRAYS

    Address   Type  Name                                          Bytes  Dimensions

  2-00000044  I*4   BUFFER                                            8  (2)
  2-0000002C  I*4   BUFFER1                                          24  (6)
  2-00000014  I*4   BUFFER2                                          24  (6)
  2-00000000  I*4   BUFFER3                                          20  (5)
  3-00000000  L*1   EMB                                             512  (0:511)
  3-00000026  L*1   EMB$B_LM_MSGTXT                                 460  (460)
  3-00000006  I*4   EMB$Q_HD_TIME                                     8  (2)

LABELS

    Address    Label        Address    Label        Address    Label         Address    Label         Address    Label         Address    Label

  0-0000004C   10         0-0000007D   25         0-0000019D   30              **       90              **       100             **       150
      **       200            **       250        1-0000000C   9000'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name                                  Type  Name                                  Type  Name

        DISK_TAPE_DRVR_MSCP_DISPATCHER              GET_CURRENT_LABEL                           INSQUE
   L*4  LIB$GET_VM                                  MOVC3                                       MOVC5
        MOVL



        Subroutine DUDRIVER_QIO (lun,emb$w_dv_func)


        include 'src$:qiocommon.for /nolist'



        byte            lun

        integer*2       emb$w_dv_func

        integer*4       qiocode(0:1,0:63)



        if (qiocode(0,0) .eq. 0) then

        qiocode(1,00) = %loc(io$_nop)
        qiocode(1,01) = %loc(io$_unload)
        qiocode(1,08) = %loc(io$_packack)

        qiocode(1,10) = %loc(io$_writecheck)
        qiocode(1,11) = %loc(io$_writepblk)
        qiocode(1,12) = %loc(io$_readpblk)

        qiocode(1,17) = %loc(io$_available)
        qiocode(1,26) = %loc(io$_setchar)
        qiocode(1,27) = %loc(io$_sensechar)

        qiocode(1,32) = %loc(io$_writelblk)
        qiocode(1,33) = %loc(io$_readlblk)
        qiocode(1,35) = %loc(io$_setmode)

        qiocode(1,39) = %loc(io$_sensemode)
        qiocode(1,48) = %loc(io$_writevblk)
        qiocode(1,49) = %loc(io$_readvblk)

        qiocode(1,50) = %loc(io$_access)
        qiocode(1,51) = %loc(io$_create)
        qiocode(1,52) = %loc(io$_deaccess)

        qiocode(1,53) = %loc(io$_delete)
        qiocode(1,54) = %loc(io$_modify)
        qiocode(1,56) = %loc(io$_acpcontrol)
        qiocode(1,57) = %loc(io$_mount)

        do 10,i = 0,63

        qiocode(0,i) = 33

        if (qiocode(1,i) .eq. 0) then
        qiocode(1,i) = %loc(qio_string)
        endif

```

```
       10      continue
               endif

               call cdrp$w_func (lun,emb$w_dv_func,
               1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))

               return
               end


PROGRAM SECTIONS

    Name                                          Bytes   Attributes

  0 $CODE                                           243   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                            8   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                          548   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 QIOCOMMON                                      1247   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                          2046

ENTRY POINTS

    Address   Type  Name

  0-00000000        DUDRIVER_QIO

VARIABLES

    Address   Type  Name                                       Address   Type  Name

 AP-00000008@ I*2   EMB$W_DV_FUNC                            2-00000200  I*4   I
  3-00000442  CHAR  IO$_ABORT                                3-0000034D  CHAR  IO$_ACCESS
  3-000003C2  CHAR  IO$_ACPCONTROL                           3-000004B3  CHAR  IO$_AVAILABLE
  3-00000297  CHAR  IO$_CLEAN                                3-00000369  CHAR  IO$_CREATE
  3-00000385  CHAR  IO$_DEACCESS                             3-00000393  CHAR  IO$_DELETE
  3-0000026D  CHAR  IO$_DIAGNOSE                             3-00000065  CHAR  IO$_DRVCLR
  3-000004CB  CHAR  IO$_DSE                                  3-000000A9  CHAR  IO$_ERASETAPE
  3-00000276  CHAR  IO$_FORMAT                               3-00000071  CHAR  IO$_INITIALIZE
  3-00000014  CHAR  IO$_LOADMCODE                            3-000003A1  CHAR  IO$_MODIFY
  3-000003E2  CHAR  IO$_MOUNT                                3-00000000  CHAR  IO$_NOP
  3-0000009D  CHAR  IO$_OFFSET                               3-000000EB  CHAR  IO$_PACKACK
  3-000000E0  CHAR  IO$_QSTOP                                3-000003EF  CHAR  IO$_RDSTATS
  3-00000421  CHAR  IO$_READCSR                              3-00000169  CHAR  IO$_READHEAD
  3-000002B6  CHAR  IO$_READLBLK                             3-0000013F  CHAR  IO$_READPBLK
  3-00000200  CHAR  IO$_READPRESET                           3-00000195  CHAR  IO$_READTRACKD
  3-0000033A  CHAR  IO$_READVBLK                             3-0000045A  CHAR  IO$_READWTHBUF
  3-00000484  CHAR  IO$_READWTHXBUF                          3-0000004D  CHAR  IO$_RECAL
  3-0000007C  CHAR  IO$_RELEASE                              3-000001AB  CHAR  IO$_REREADN
  3-000001B8  CHAR  IO$_REREADP                              3-000000CA  CHAR  IO$_RETCENTER
  3-000002E6  CHAR  IO$_REWIND                               3-000002C9  CHAR  IO$_REWINDOFF
  3-000000FC  CHAR  IO$_SEARCH                               3-00000024  CHAR  IO$_SEEK
  3-00000231  CHAR  IO$_SENSECHAR                            3-00000309  CHAR  IO$_SENSEMODE
```

```
                                                                                M 1
DUDRIVER_QIO                                                               16-Sep-1984 00:20:36    VAX-11 FORTRAN V3.4-56                Page  16
                                                                            5-Sep-1984 13:54:29    DISK$VMSMASTER:[ERF.SRC]DUTUDRIVR.FOR;1

  3-0000021D  CHAR IO$_SETCHAR                          3-000003B8  CHAR IO$_SETCLOCK
  3-00000088  CHAR IO$_SETCLOCKP                        3-000002DD  CHAR IO$_SETMODE
  3-000002ED  CHAR IO$_SKIPFILE                         3-000002FA  CHAR IO$_SKIPRECORD
  3-00000029  CHAR IO$_SPACEFILE                        3-0000010E  CHAR IO$_SPACERECORD
  3-000003D7  CHAR IO$_STARTDATA                        3-000000B4  CHAR IO$_STARTDATAP
  3-00000037  CHAR IO$_STARTMPROC                       3-0000020F  CHAR IO$_STARTSPNDL
  3-00000059  CHAR IO$_STOP                             3-0000000D  CHAR IO$_UNLOAD
  3-0000046B  CHAR IO$_WRITEBUFNCRC                     3-0000011E  CHAR IO$_WRITECHECK
  3-000001E4  CHAR IO$_WRITECHECKH                      3-000003FF  CHAR IO$_WRITECSR
  3-00000153  CHAR IO$_WRITEHEAD                        3-000002A2  CHAR IO$_WRITELBLK
  3-00000247  CHAR IO$_WRITEMARK                        3-00000314  CHAR IO$_WRITEOF
  3-0000012A  CHAR IO$_WRITEPBLK                        3-000001C9  CHAR IO$_WRITERET
  3-0000017E  CHAR IO$_WRITETRACKD                      3-00000326  CHAR IO$_WRITEVBLK
  3-00000448  CHAR IO$_WRITEWTHBUF                      3-00000257  CHAR IO$_WRTTMKR
 AP-0000004@ L*1  LUN                                   3-000004A1  CHAR QIO_STRING

ARRAYS

    Address   Type  Name                  Bytes  Dimensions

  2-00000000  I*4   QIOCODE                 512  (0:1, 0:63)

LABELS

    Address   Label

      **      10

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name             Type  Name

        CDRP$W_FUNC       I*4  LIB$EXTZV
        Subroutine TUDRIVER_QIO (lun,emb$w_dv_func)

        include 'src$:qiocommon.for /nolist'


        byte            lun

        integer*2       emb$w_dv_func

        integer*4       qiocode(0:1,0:63)



        if (qiocode(0,0) .eq. 0) then

        qiocode(1,00) = %loc(io$_nop)
        qiocode(1,01) = %loc(io$_unload)
        qiocode(1,08) = %loc(io$_packack)

        qiocode(1,10) = %loc(io$_writecheck)
        qiocode(1,11) = %loc(io$_writepblk)
        qiocode(1,12) = %loc(io$_readpblk)

        qiocode(1,17) = %loc(io$_available)
        qiocode(1,21) = %LOC(io$_dse)
        qiocode(1,26) = %loc(io$_setchar)

        qiocode(1,27) = %loc(io$_sensechar)
        qiocode(1,32) = %loc(io$_writelblk)
        qiocode(1,33) = %loc(io$_readlblk)

        qiocode(1,35) = %loc(io$_setmode)
        qiocode(1,39) = %loc(io$_sensemode)
        qiocode(1,48) = %loc(io$_writevblk)

        qiocode(1,49) = %loc(io$_readvblk)
        qiocode(1,50) = %loc(io$_access)
        qiocode(1,51) = %loc(io$_create)

        qiocode(1,52) = %loc(io$_deaccess)
        qiocode(1,53) = %loc(io$_delete)
        qiocode(1,54) = %loc(io$_modify)

        qiocode(1,56) = %loc(io$_acpcontrol)
        qiocode(1,57) = %loc(io$_mount)

        do 10,i = 0,63

        qiocode(0,i) = 33

        if (qiocode(1,i) .eq. 0) then
        qiocode(1,i) = %loc(qio_string)
        endif
```

```
      10        continue
                endif

                call cdrp$w_func (lun,emb$w_dv_func,
                1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))

                return
                end
```

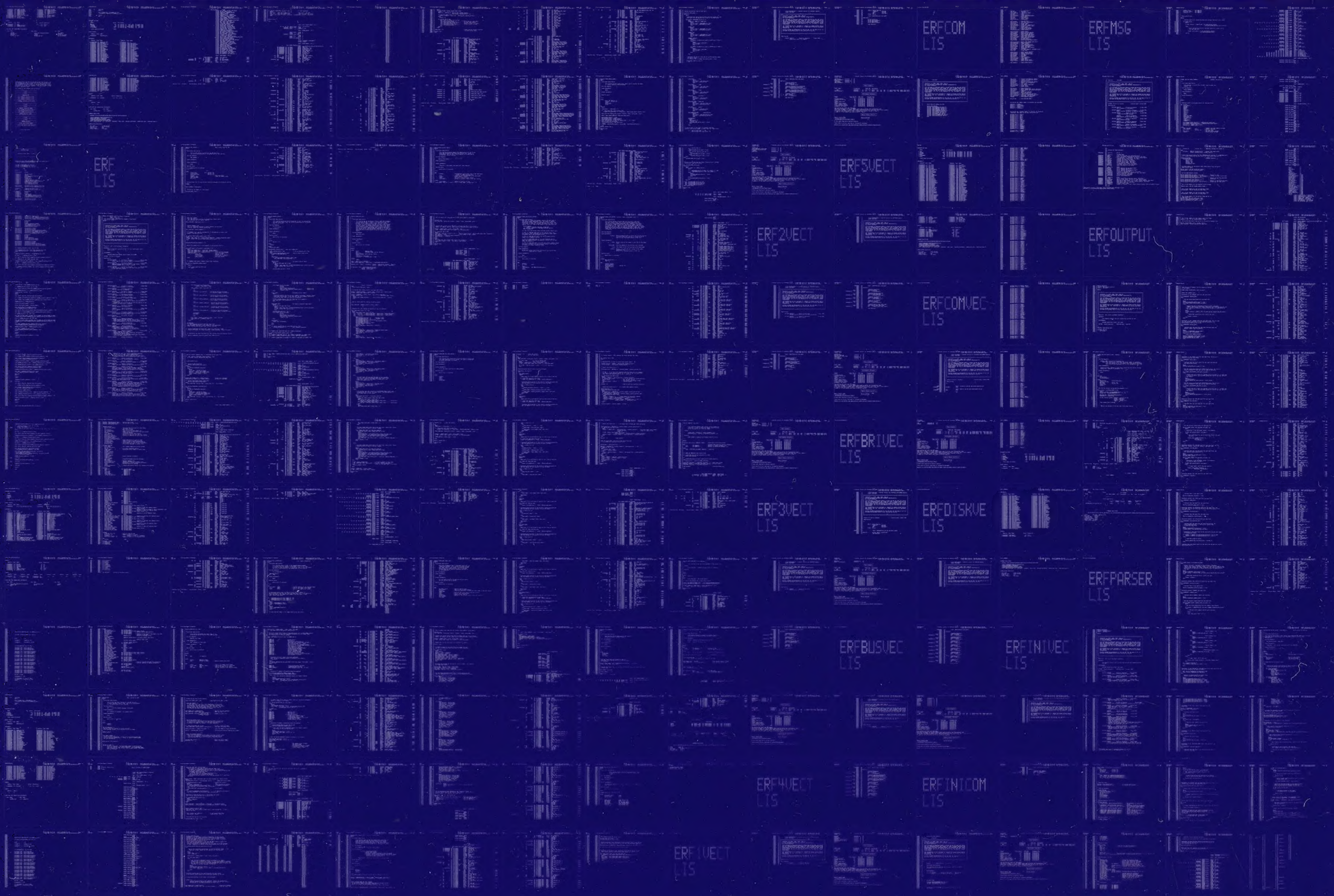
```
PROGRAM SECTIONS

    Name                                             Bytes   Attributes

  0 $CODE                                              250   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                               8   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                             548   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 QIOCOMMON                                         1247   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                             2053


ENTRY POINTS

    Address   Type  Name

  0-00000000        TUDRIVER_QIO


VARIABLES

    Address   Type  Name                                          Address   Type  Name

 AP-00000008@ I*2   EMB$W_DV_FUNC                              2-00000200  I*4   I
  3-00000442  CHAR  IO$_ABORT                                  3-0000034D  CHAR  IO$_ACCESS
  3-000003C2  CHAR  IO$_ACPCONTROL                             3-000004B3  CHAR  IO$_AVAILABLE
  3-00000297  CHAR  IO$_CLEAN                                  3-00000369  CHAR  IO$_CREATE
  3-00000385  CHAR  IO$_DEACCESS                               3-00000393  CHAR  IO$_DELETE
  3-0000026D  CHAR  IO$_DIAGNOSE                               3-00000065  CHAR  IO$_DRVCLR
  3-000004CB  CHAR  IO$_DSE                                    3-000000A9  CHAR  IO$_ERASETAPE
  3-00000276  CHAR  IO$_FORMAT                                 3-00000071  CHAR  IO$_INITIALIZE
  3-00000014  CHAR  IO$_LOADMCODE                              3-000003A1  CHAR  IO$_MODIFY
  3-000003E2  CHAR  IO$_MOUNT                                  3-00000000  CHAR  IO$_NOP
  3-0000009D  CHAR  IO$_OFFSET                                 3-000000EB  CHAR  IO$_PACKACK
  3-000000E0  CHAR  IO$_QSTOP                                  3-000003EF  CHAR  IO$_RDSTATS
  3-00000421  CHAR  IO$_READCSR                                3-00000169  CHAR  IO$_READHEAD
  3-00000286  CHAR  IO$_READLBLK                               3-0000013F  CHAR  IO$_READPBLK
  3-00000200  CHAR  IO$_READPRESET                             3-00000195  CHAR  IO$_READTRACKD
  3-0000033A  CHAR  IO$_READVBLK                               3-0000045A  CHAR  IO$_READWTHBUF
  3-00000484  CHAR  IO$_READWTHXBUF                            3-0000004D  CHAR  IO$_RECAL
  3-0000007C  CHAR  IO$_RELEASE                                3-000001AB  CHAR  IO$_REREADN
  3-000001B8  CHAR  IO$_REREADP                                3-000000CA  CHAR  IO$_RETCENTER
  3-000002E6  CHAR  IO$_REWIND                                 3-000002C9  CHAR  IO$_REWINDOFF
  3-000000FC  CHAR  IO$_SEARCH                                 3-00000024  CHAR  IO$_SEEK
  3-00000231  CHAR  IO$_SENSECHAR                              3-00000309  CHAR  IO$_SENSEMODE
  3-0000021D  CHAR IO$_SETCHAR                    3-000003B8  CHAR IO$_SETCLOCK
  3-00000088  CHAR IO$_SETCLOCKP                  3-000002DD  CHAR IO$_SETMODE
  3-000002ED  CHAR IO$_SKIPFILE                   3-000002FA  CHAR IO$_SKIPRECORD
  3-00000029  CHAR IO$_SPACEFILE                  3-0000010E  CHAR IO$_SPACERECORD
  3-000003D7  CHAR IO$_STARTDATA                  3-000000B4  CHAR IO$_STARTDATAP
  3-00000037  CHAR IO$_STARTMPROC                 3-0000020F  CHAR IO$_STARTSPNDL
  3-00000059  CHAR IO$_STOP                       3-0000000D  CHAR IO$_UNLOAD
  3-0000046B  CHAR IO$_WRITEBUFNCRC               3-0000011E  CHAR IO$_WRITECHECK
  3-000001E4  CHAR IO$_WRITECHECKH                3-000003FF  CHAR IO$_WRITECSR
  3-00000153  CHAR IO$_WRITEHEAD                  3-000002A2  CHAR IO$_WRITELBLK
  3-00000247  CHAR IO$_WRITEMARK                  3-00000314  CHAR IO$_WRITEOF
  3-0000012A  CHAR IO$_WRITEPBLK                  3-000001C9  CHAR IO$_WRITERET
  3-0000017E  CHAR IO$_WRITETRACKD                3-00000326  CHAR IO$_WRITEVBLK
  3-00000448  CHAR IO$_WRITEWTHBUF                3-00000257  CHAR IO$_WRTTMKR
 AP-00000004@ L*1  LUN                            3-000004A1  CHAR QIO_STRING

ARRAYS

    Address  Type  Name                 Bytes  Dimensions

  2-00000000  I*4  QIOCODE                512  (0:1, 0:63)

LABELS

    Address   Label

      **      10

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name

        CDRP$W_FUNC        I*4  LIB$EXTZV

COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:DUTUDRIVR/OBJ=OBJ$:DUTUDRIVR MSRC$:DUTUDRIVR

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19

COMPILATION STATISTICS

  Run Time:           11.00 seconds
  Elapsed Time:       28.53 seconds
  Page Faults:        373
  Dynamic Memory:     199 pages
```

0147 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0148 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



ERFCOM
LIS

ERFMSG
LIS

ERF
LIS

ERF5VECT
LIS

ERF2VECT
LIS

ERFOUTPUT
LIS

ERFCOMVEC
LIS

ERFBRIVEC
LIS

ERF3VECT
LIS

ERFDISKVE
LIS

ERFPARSER
LIS

ERFBUSVEC
LIS

ERFINIVEC
LIS

ERF4VECT
LIS

ERFINICOM
LIS

ERF1VECT
LIS